MASTERFILE 464 PROGRAM EXTENSIONS

for the AMSTRAD CPC 464/664

CAMPBELL SYSTEMS

CS

--- Contents ---

Campbell Systems
57 Trap's Hill
Loughton
Essex IG10 1TD

Tel: 01-508 5058

MASTERFILE 464 PROGRAM EXTENSIONS

INTRODUCTION

For those users who wish to link MASTERFILE 464 with other programs, or who wish to perform special processing on their MASTERFILE data, the following two software components are offered:

a) MASTEXPT: Allows file data to be routed to other programs.

b) MASTPROC: Allows the user to perform BASIC processing on a file.

MASTEXPT is used to extract data from a file left in memory by MASTERFILE 464, and to package it in a form suitable for other programs, such as TASWORD.

MASTPROC is a file "tool-kit" which can create/update/extend/retrieve a file. It is called from within Basic written by the user, and is therefore suitable only for those who have a grasp of Amstrad BASIC. One application of MASTPROC is to run in conjunction with MASTEXPT to merge files, and for this we have provided the MERGE program as an illustration of MASTPROC use.

CAVEAT

We must stress that MASTPROC is not for very user, and whilst we will always help out with MASTERFILE 464 queries, we cannot gaurantee to help users debug their MASTPROC BASIC code.

CASSETTE/DISC CONTENTS

The MASTERFILE 464 Program Extension pack contains the following components:

MASTEXPT machine code.

MERGE Basic program example of MASTPROC use.

MASTPROC machine code. Loaded by MERGE or other User Basic.

MEMORY ORGANISATION

The memory map below shows have the various program components fit in relation to the file. The map is not drawn to scale.

```
                 MASTERFILE 464 / MASTEXPT / MASTPROC RAM Map
:-----------------------------------------------------------------------:
: &AC00(approx)-&FFFF Stack, firmware data and screen map               :
:-----------------------------------------------------------------------:
: Highest address depends on hardware configuration; value in "HM".     :
:                                                                       :
:                                                                       :
:   MASTERFILE 464 file space. File is built from low address upwards.  :
:                                                                       :
:                                                                       :
: Lowest address approx &2690                                           :
:-----------------------------------------------------------------------:
: MASTCODE main          : MASTEXPT m/c code      : MASTPROC m/c code   :
: program logic          :                        :------- HIMEM = &235F -:
: (approx 9K)            :                        :                     :
:                        :                        : MASTPROC User's Basic :
:-------------------------------- HIMEM = &37F ---: (approx 8K)         :
: MASTLOAD Basic                                  :                     :
:-----------------------------------------------------------------------:
: 0000-&170 Firmware use                                                :
:-----------------------------------------------------------------------:
```

MASTEXPT: MASTERFILE 464 DATA EXPORT PROGRAM

The files created by MASTERFILE 464 are not suitable for input directly into other programs, but MASTERFILE 464 file data can be "exported" to other programs by means of the export utility, MASTEXPT. A particular use of this utility is to send data to the TASWORD/AMSWORD word-processor program. Other Basic programs may also read MASTEXPT output, using OPENIN and INPUT #9 etc. Our "MERGE" example does this.

The utility works by processing a file which is left in the computer's memory by MASTERFILE 464. One uses MASTERFILE 464 to select the records from which data is to be extracted. Then the utility is loaded as follows:

Press B at main menu to return to BASIC.

LOAD"MASTEXPT" [ENTER] to load the export utility, overwriting MASTCODE.

Use ¦TAPE or ¦DISC etc as required for the export file to be created.

GOTO 100 [ENTER] to give control to the utility.

The utility asks how the data is to be packaged, and requires simple Y or N responses. There are four questions concerned with package style, and the first is:

Data identifiers to be sent ? Y/N

Your data can be exported with or without identifiers in front of each field. An identifier is two characters: ampersand and data reference, e.g. "&D". If data is being exported to TASWORD/AMSWORD for mail-merge purposes, then you must reply Y. The next question is:

LF as well as CR ? Y/N

Every field and record is suffixed with a CR byte (decimal 13). If data is being sent to TASWORD/AMSWORD then every CR must also be followed by a LF byte which is decimal 10. So for TASWORD/AMSWORD, reply Y. Otherwise, reply N. The next question is:

Edit line breaks etc ? Y/N

Reply N if the data is to be routed back to MASTERFILE, for example as part of a file merge process. Otherwise, you may wish to process line-break characters and sort key prefix (\) characters - in which case reply Y. For TASWORD/AMSWORD use reply with Y. The next question is:

Soft EOF to end ? Y/N

The export file can be terminated with the "soft end-of-file" character, decimal 26, if required. Reply Y or N, but note that for TASWORD/AMSWORD use, reply N. For MASTPROC use, reply Y.

Next, the utility asks which data fields are to be extracted. You can select just one field, or any number of fields, in any order. Further, you can specify substitute data where a field is absent in a record. The prompt is:

Give reference of data to extract, or ENTER if no more

Reply with the data reference of the target field. This must be a reference which is present in the file's list of data names. Next you are asked:

Default text if no data:

Enter any text as required, and press [ENTER]. Use [ENTER] by itself if no such text is required. Thus, for any records for which the extracted data is missing, the default text is substituted. Examples of default text might be "The Manager", "Unknown".

Data names and default texts are listed on the screen, and the utility keeps asking for more references until [ENTER] is pressed. Then, one more response is required, which is the name of the file to be created:

Give file name for output:

Reply with a valid file name, 1-8 characters. The file is then written, and on completion the utility beeps and displays:

*** File Complete ***

(This may vanish instantly if Basic next does MODE or CLS.)

The utility finishes with an exit to Basic. If another export file is required from the same selected MASTERFILE 464 file, then you can GOTO 100 and start again.

OUTPUT FILE STRUCTURE

The output file is ASCI, each field and record terminated with a CR. (and LF if required.) End of file is terminated with Soft EOF (decimal 26) if required. Within each record, each data field is optionally prefixed with ampersand and data reference, and is terminated with CR (and LF if required.)

If you opted for editing of line-breaks etc, then within each data field, any line-break characters are converted to CR (and LF if required.). Also, any "\" characters are converted to spaces. "\" is used by MASTERFILE 464 to specify an embedded sort key.

The order in which data appears in each record is the order in which data references were specified to the utility. Any fields which were absent in the file and for which no default text is given, are present even though empty.

Only records currently SELECTED by MASTERFILE 464 will produce records to the export file.

Data absent in a record is sought from a matching PARENT record.

TASWORD/AMSWORD

For a file to be read by this program, the option to follow each CR with LF must be chosen. The export file can then be read directly by TASWORD/AMSWORD.

If the export file is for mail-merge use in TASWORD 464-D, then the option to send data identifiers must also be chosen. The package options are therefore: Y Y Y N.

An exported file loaded into TASWORD/AMSWORD shows with each data field starting a fresh line, and there is an extra blank line between each record.

FILE CAPACITY

MASTEXPT output file buffer is allocated without placing any constraint on the current file size, which can therefore be as full as the main MASTERFILE program allows.

SAVING THE UTILITY

The export utility may be saved to tape or disc, as follows:

```
SAVE"MASTEXPT",B,&380,&780 [ENTER]
```

If doing this without first loading MASTERFILE 464, then you need to:

```
OPENOUT"X":MEMORY &37F:LOAD "MASTEXPT"  [ENTER]
```

FILE MERGE

One use of MASTEXPT is in conjunction with MASTPROC, whereby two files can be united into a single combined file for use by MASTERFILE proper. The "primary" file is the one whose data names and report formats are to be kept. The "secondary" file is the one from which selected records are to be appended to the primary file. The secondary file must be processed by MASTEXPT. In other words, data to be added from one file onto another must first be exported via MASTEXPT.The data packaging options should be Y N N Y, all data references must be given, and no default text should be given.

The MERGE Basic is listed in the MASTPROC guide, which follows.

MASTPROC: MASTERFILE 464 FILE PROCESSOR UTILITY

The File Processor Utility, hereafter referred to as "MASTPROC", allows a user with some knowledge of Amstrad BASIC to process MASTERFILE files in a way not possible with MASTERFILE alone. For example, MASTPROC can be used to:

a) Create a new file directly from another source.

b) Compute and store new data in each record, derived from other data. (e.g. compute value of a stock file by multiplying quantity by unit cost.)

c) Erase/insert/amend data items in each record.

d) Erase records.

e) Insert new records.

f) Retrieve data and create an external file to be read by other programs.

To use MASTPROC the user must learn a little about MASTERFILE file structure, and about CPC464 memory management. A workable knowledge of BASIC is also assumed. It follows that MASTPROC is not suitable for all users: but for those who can absorb the technical information in this guide, MASTERFILE can be expanded into a very powerful information processor.

MASTERFILE File Structure

A file consists of control data, data names, report formats, and file data. We need only be concerned with file data, which is organised into records. Each record has a one-byte header of either &EC (unselected) or &C0 (selected). We are concerned only with selected records, since unselected records are not made accessible to MASTPROC.

Each record consists of data items, in any order, and each item comprises:

```
1st byte: Data Reference, in the range !(&21) to ↑(&5E)
2nd byte: Length in binary of the following data
3rd byte etc: Data, 1 to 240 characters
```

Please note that the 2nd byte is not necessarily a legal printable character. Data items are passed in undimensioned strings, and for such an item in D$:

a) Data reference is LEFT$(D$,1)

b) Data length is ASC(MID$(D$,2,1))

c) Data text is MID$(D$,3,ASC(MID$(D$,2,1)))

Data text may also be represented by MID$(D$,3) - a simpler expression, but not strictly correct if D$ is longer than necessary. But if padded with spaces, one can use this expression to retrieve and convert numeric data, viz:

```
VAL(MID$(D$,3)).
```

Synopsis of operation

The user writes his file processing logic in Basic, which will include CALLs to the MASTPROC machine code. The MASTPROC machine code module must be loaded, then the MASTERFILE file (unless one is to be created from scratch). The Basic is RUN and on completion of processing, the file can be re-saved ready for loading into MASTERFILE 464.

WRITING MASTPROC BASIC

The program must start as follows:

```
10 HM=HIMEM:ZZ=&2360:MEMORY ZZ-1:LOAD"MASTPROC
20 NEWF=ZZ:INIT=ZZ+3:GETR=ZZ+6:GETD=ZZ+9:PUTD=ZZ+12:ERAD=ZZ+15:
   ERAR=ZZ+18: INSR=ZZ+21:WRFL=ZZ+24
```

Line 10 above sets HM to the initial HIMEM, alters HIMEM to &235F, and loads the file access module, named MASTPROC.

Line 20 sets up 9 symbolic entry points to be CALLed by the rest of the program. The actual names used are not critical, but we suggest you use our names.

Before loading your Basic, a full reset is recommended in order to guarantee that HM is set up correctly in line 10.

The rest of the program will be up to you, and will probably start with a LOAD of a file created by MASTERFILE 464. There is no need to give a load address, as the file will load at the same address as where it was when saved. The MASTPROC functions are described below. Our illustrations use variable names D$, IND$, etc. but you may use different names, provided you do so consistently.

INIT: Initialise after loading a file

```
      IND$=CHR$(32):CALL INIT,HM,@IND$:IF IND$="E" THEN ...error...
```

IND$ is a one-byte indicator, set to "E" only if INIT fails to locate the file in memory. INIT must be called before a loaded MASTERFILE file can be further processed.

NEWF: Initialise a new file

```
      CALL NEWF,HM
```

This call establishes a new file, ready for record insertion. A file created this way will be without any data names or formats, but these can be added later by MASTERFILE proper. It is unlikely that both NEWF and INIT would be used in the same program.

GETR: Access a record

Only selected data records can be accessed. Although it would be usual to access records sequentially, one can instead read them in any order. The records are not physically numbered, it is just that record number 5 is the 5th selected record in the file. Access of a record is necessary before one can use functions GETD, PUTD, ERAD, ERAR. An accessed record is "current". The call is:

```
      CALL GETR,N,@IND$
```

where N is a numeric variable containing the number of the record to be accessed, or it can be an explicit number. IND$ is set to "E" on exit if the target record is not found - i.e. there are fewer than "N" selected records.

To access a file sequentially:

```
      N=1:IND$=" "
      WHILE 1
      CALL GETR,N,@IND$:N=N+1:IF IND$="E" GOTO xxxx
      WEND
```

(Line xxxx gets control on end-of-file.)

GETD: Get data from current record

To get data from a record, you must initialise a string variable to receive it, and of sufficient length. If it is not long enough, the data will be truncated. For example, to retrieve data whose data reference is "N", and where the data is not expected to be longer than 20 characters:

```
D$="N"+SPACE$(21): CALL GETD,@D$
```

Note that D$ is used both to specify the data reference and to receive the retrieved data item, including reference and data length. If the data is absent from the record then the data length is set to zero and can be tested thus:

```
IF MID$(D$,2,1)=CHR$(0) THEN ...not found...
```

PUTD: Insert/replace data into current record

An item can be inserted into the current record, replacing any item of the same data reference. The item is passed in a string variable, including reference and length byte - although the latter is not used as the data length is taken from the LEN attribute of the passed string. For example, to store data "J Smith" with data reference "N":

```
D$="N.J Smith": CALL PUTD,@D$
```

Any invalid data characters (outside the range &20-&7E) are altered to "*" and any leading spaces are dropped. You may notice that the second byte of the literal in the D$ assignment gets altered. This is because of CPC464 Basic's use of the statement text directly to hold string data: PUTD stores the binary data length into this second byte. This is harmless.

ERAD: Erase data from the current record

To erase a data item from the current record, the target data reference must be passed. For example, to erase data whose reference is "R":

```
D$="R": CALL ERAD,@D$
```

If there is no matching data item in the current record, then no action is taken.

ERAR: Erase the current record

To erase the current record:

```
CALL ERAR  (there are no parameters)
```

Notice that after erasing the nth record, the next selected record in sequence is still the nth, since GETR counts from the beginning of the file every time. After an ERAR call, there is no "current" record until another GETR is used.

INSR: Insert a new record

A new record can be inserted, either immediately after the current record or if there is no current record then the new record is placed at the end of the file. The new record becomes the current record and has "selected" status. Until PUTD is used, a new record consists of just a header byte.

```
CALL INSR  (there are no parameters)
```

WRFL: Write the file

The file can be written to tape or disc, and the file name must be passed, for example as follows:

```
     N$="SALESREV":CALL WRFL,@N$
```

This works just as if the file had been saved using MASTERFILE 464 proper. No extra buffer space is required.

Buffer Space

The LOAD of MASTPROC causes BASIC to allocate a 4K buffer area, reducing the memory left to Basic to about 4K. In fact, BASIC does not use this buffer area unless an OPENIN or OPENOUT is given in order to use INPUT #9 or PRINT #9. So, unless your BASIC program does use OPENIN or OPENOUT then you can make BASIC allocate the buffer area at the top of RAM, leaving more room for your program. You do this by opening a dummy file before the MEMORY statement is reached, viz:

```
     5 OPENOUT"dummy"
```

The effect of the MEMORY statement later is to fix this high buffer area, and thus Basic is left with 8K for itself.

Error Conditions

Errors are warned audibly, for example using GETD when there is no current record, or when there an insertion is impossible due to lack of RAM space. We do therefore urge that you do not run with the sound turned right down.

Saving MASTPROC

The machine code module can be saved via:

```
    SAVE "MASTPROC",B,&2360,&328
```

Please note that as with MASTCODE, this should only be done before any file is loaded or started.

Example 1

Suppose that a stock file contains in each selected record the following:

    Data reference Q = quantity in stock
    Data reference C = cost per unit

There may be other data, such as stock code, description, re-order level etc., but we are only concerned with Q and C. We wish to add to each record:

    Data reference V = Q*C i.e. value being quantity x unit cost.

MASTPROC Basic would be as follows (we only show the central part):

```
   30 IND$=CHR$(32):CALL INIT,HM,@IND$:IF IND$="E" GOTO xxxx
   40 N=1: WHILE 1
   50 CALL GETR,N,@IND$:N=N+1:IF IND$="E" GOTO yyyy
   60 D$="Q"+SPACE$(20):CALL GETD,@D$:VQ=VAL(MID$(D$,3))
   70 D$="C"+SPACE$(20):CALL GETD,@D$:VC=VAL(MID$(D$,3))
   80 D$="V."+STR$(VQ*VC):CALL PUTD,@D$
   90 WEND
```

[xxxx is the line to get control if line 30 gives an error.
  Line yyyy gets control at end-of-file.]

Since all MASTERFILE data is in character format, we use VAL function to convert data into numeric form ready for arithmetic. Conversely, we use STR$ function to convert back to character form before using PUTD. Note that STR$ of a positive number starts with a space; but PUTD will trim this off.

Example 2

A file "merge" operation appends data records from one file onto another file, and this can be done in the following stages:

a) Starting with the file to be appended, use MASTEXPT to export a file containing all possible data fields, with data identifiers, no LF suffix, no edit of line-break, and logical EOF. (i.e. Packaging options Y N N Y.). This will create the "export" file.

b) Then use MASTPROC as set out below, and load the "primary" as saved by MASTERFILE proper, and then the "export" file as created by MASTEXPT.

```
 10 'Merge a file with exported data
 20 '---------------------------------
 30 hm=HIMEM:ZZ=&2360:MEMORY ZZ-1:LOAD"MASTPROC
 40 NEWF=ZZ:INIT=ZZ+3:GETR=ZZ+6:GETD=ZZ+9:PUTD=ZZ+12:ERAD=ZZ+15:
    ERAR=ZZ+18: INSR=ZZ+21: WRFL=ZZ+24
 50 CLS:PRINT"Name of primary MASTERFILE file to load ";:INPUT n$:
    LOAD n$
 60 PRINT"Name of export file to append ";:INPUT n$:OPENIN n$
 70 IND$=CHR$(32): CALL INIT,HM,@IND$:CALL INSR
 80 WHILE NOT EOF
 90 LINE INPUT #9,D$
100 IF LEN(D$)=0 THEN CALL INSR
110 IF LEN(D$)>2 THEN MID$(D$,1,1)=MID$(D$,2):CALL PUTD,@D$
120 WEND
130 CALL ERAR
140 PRINT"Name of new merged file ";:INPUT n$:CALL WRFL,@n$
```

Line 70 initialises the primary file and appends the first new record header onto the end of the file. Line 90 reads the file created by MASTEXPT. Line 100 inserts a new current record each time end-of-record (null data) is read. Line 110 rebuilds the data fields and inserts them into the file. Empty data items, consisting of just ampersand/ref, are dropped. Line 130 erases the last over-shoot record header. Note that unlike the MASTEXPT stage, MASTPROC does not need to know which data references are present.

This program is also on the delivered cassette/disc, as "MERGE".

Suggested Uses

MASTPROC uses are limited only by the imagination, and 8K. Possibly most uses will revolve around computation of numeric data, as in our 1st example. A variation might be to generate a cross-total and average of exam marks for each pupil. Another variation would be to flag records in a stock file where quantity in stock falls below recorder level.

Another kind of use might be as a streamlined file update. MASTERFILE 464 Display/Alter is fine for small-volume changes, but if you need to update many records, then this can get tedious. MASTPROC can present each record in turn, prompt for fields to be revised, and advance to the next record, until all selected records are done.

MASTPROC is also the ideal tool for converting data from other programs into MASTERFILE format, either appending to a file (as in the MERGE example) or by originating a new file (using NEWF). It can also work in the opposite direction to export data in a way that is not possible with MASTEXPT.

MASTPROC can also match two files (one in RAM, the other via MASTEXPT), as it can scan the RAM file for a match against every record in the other file.